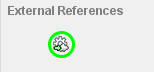
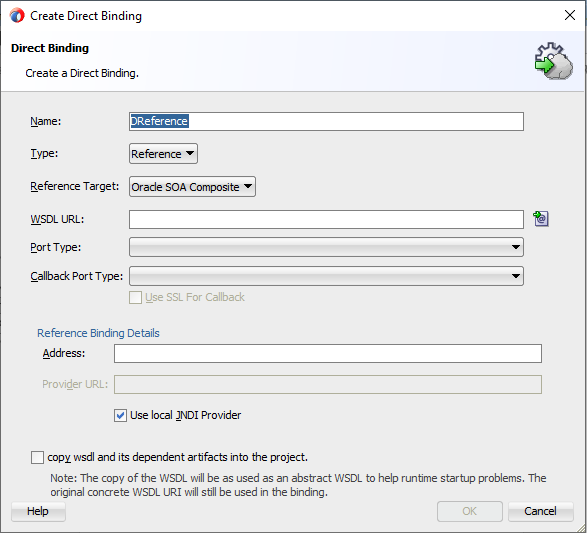
**Creando conexión con servicio externo (Todo Service Bus)**

En esta serie de tutorial

1. Cuando se hizo el **tutorial 2** del “**Service Bus**”, el mismo fue expuesto mediante el uso del conector “**sb**”. Este no presenta una interfaz externa como “**Http**”, pero permite la conexión interna de distintas aplicaciones.
2. Este tutorial parte del creado anteriormente.
3. Primero vamos al composite ().
4. En el menú de componentes buscamos **“Direct” []** y lo arrastramos hacia la sección de “**External References**”.



1. Esto nos va a levantar la siguiente ventana**:**



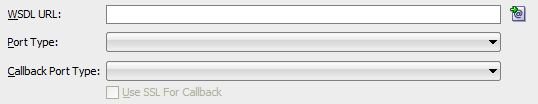
* 1. **Name:** Nombre que le pondremos al componente.
  2. **Type:** Tipo de conexión a utilizar. Esta permite seleccionar entre **“Service”** y **“Reference”.** En nuestro caso vamos a usar **“Reference”** ya que queremos apuntar a un servicio ya desplegado.



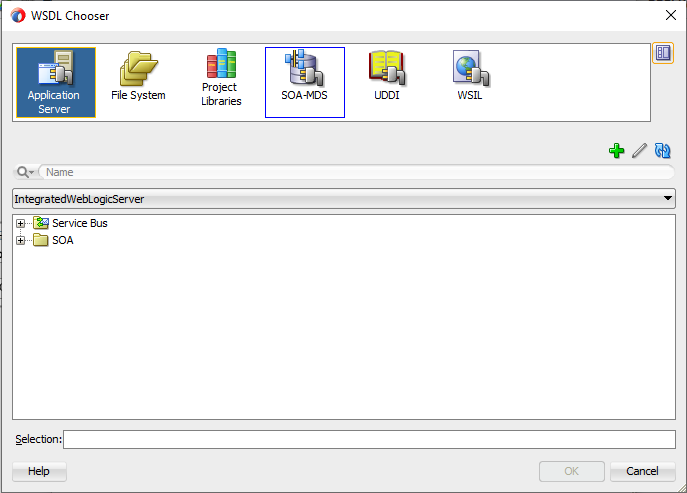
* 1. **Reference Target:** El objetivo de nuestra referencia. Aquí podemos decir sí vamos a utilizar un composite externo para obtener datos o un bus de servicio ya desplegado. En nuestro caso vamos a seleccionar el “**Oracle Service Bus**”



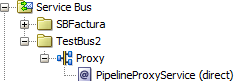
* 1. Después tenemos la referencia al **“WSDL”** a utilizar**.**



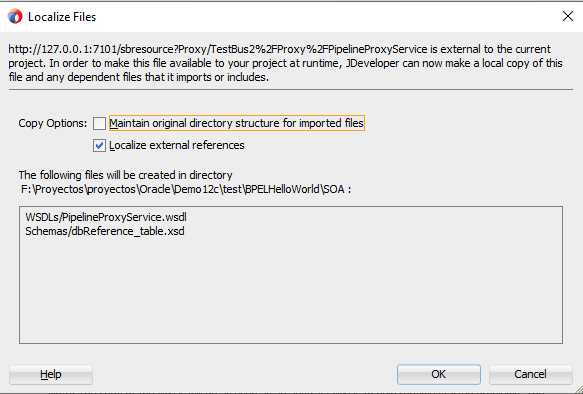
* + 1. Presionamos [] para que nos abra la ventana de selección del “**WSDL**”.



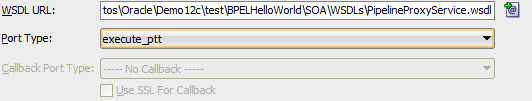
* + 1. En este caso queremos seleccionar un servicio ya desplegado, por lo que dentro de [] desplegamos la opción de [].
    2. Dentro buscamos el “**Service Bus**” desplegado.



* + 1. Seleccionamos el archivo “**WSDL**” que queremos utilizar y presionamos [**OK**]
    2. Ahora pasamos a una nueva ventana que nos permitirá importar los archivos propios del “**Service Bus**” a utilizar.



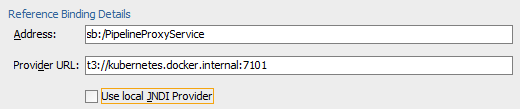
* + 1. Podemos dejarlo como viene por defecto y presionamos [**OK**].
    2. Con todas estas modificaciones, nos queda la sección de la siguiente forma:



* + 1. En caso de que tengamos varios “**Port Type**” creados, es posible elegir uno mediante:



* + 1. Por último, tenemos la conexión a realizar con el servicio importado.



* + 1. Presiona [**OK**].
  1. Esto nos crea el siguiente componente dentro del “**composite**” del “**BPEL**”

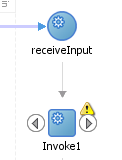
1. Ahora nos toca vincular el “BPEL” con la referencia externa creada.



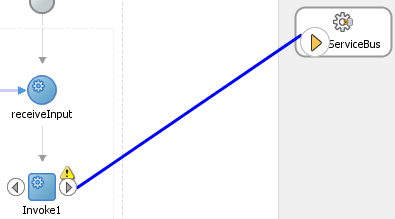
1. Ahora tenemos que vincularlo en el archivo de “**BPEL**”. El servicio se encuentra en la secci



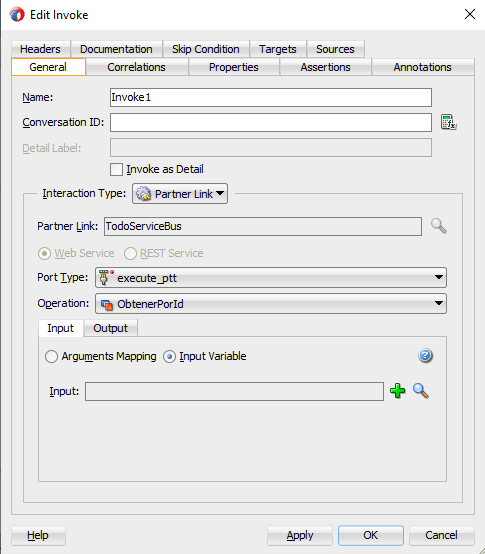
1. Para vincular el flujo del “**BPEL**” con el “**Partner Link**” del servicio creado, debemos añadir un nodo “**Invoke**” [] en la parte del flujo que queramos.



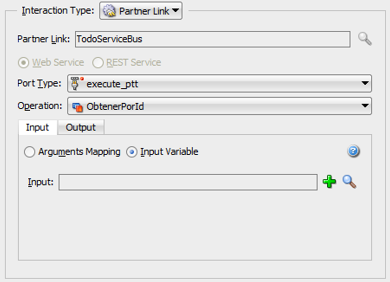
1. El “**Invoke**” pide que le añadas una conexión a un servicio. Por lo que mediante la [] creamos la conexión con el “**external service**”.



1. Esto nos levanta una ventana para la selección de las opciones del invocador.



* 1. **Name**: Nombre que se le pondrá al componente de invocación.
  2. **Conversation** **ID**: Esto podemos dejarlo vacío, pero es el ID que se le asignaría al “**Invoke**”.
  3. **Interaction Type**:



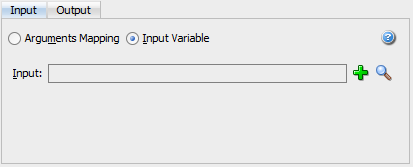
* + 1. Esta parte es donde se muestra la interacción del “**invoke**” creado con el “**partern link**” vinculado.
    2. : Nos permite seleccionar con quien nos vamos a vincular.
       1. Existen dos opciones:
          1. Partner Link
          2. Event
    3. **Partner Link**: Aquí seleccionamos el servicio a invocar. En este caso no se puede modificar, ya que estamos haciendo una asignación directa.



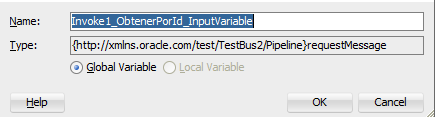
* + 1. **Port Type y Operation**: Estos dos [**select**], nos permiten buscar el tipo de puerto, del servicio vinculado, que vamos a utilizar (los puertos creados en el **WSDL**) y la operación a realizar (definida en el **WSDL**).



* + 1. Por último, tenemos las variables:



* + - 1. Aquí podemos definir las variables de entrada y salida que va a utilizar el “**Invoke**”.
         1. La de salida no es obligatoria si lo único que se piensa hacer es crear una llamada unidireccional; o sea, pasar un mensaje a otro servicio y no esperar respuesta.
      2. En ambos casos funciona igual. Podemos utilizar el símbolo de () para crear una nueva variable o () para buscar una variable ya creada.
      3. En este caso vamos a crear una nueva, por lo que presionamos ().



* + - * 1. **Name**: Nombre que va a recibir la variable.
        2. **Type**: El tipo de variable. En este caso es una composición del “**namespace”** de llamada al servicio externo y “**requestMessage**”. Este último nombre, hace referencia a que es un mensaje de petición hacia ese servicio. En caso del output () podemos ver cómo cambia a “**replayMessage**”.

El “**Type”** no es modificable.

* + - * 1. : Las variables tienen dos formas de declararse, Global o Local; piensa de la siguiente forma:

Global: Variable declarada el principio de la clase como:

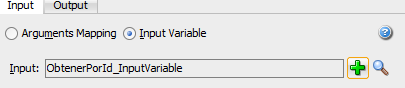
private int global;

Local: Variable declarada dentro de una función:

function test() { const local = “Test”}

En este caso, el global lo da su uso en todo el flujo del BPEL y el Local, su uso dentro de un “**scope**”.

* + - * 1. Presionamos [**OK**]
      1. Esto nos autocompleta la variable de entrada de la siguiente forma:



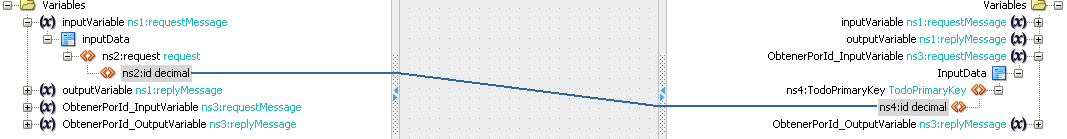
* + - 1. Hacemos el mismo procedimiento para la variable de salida “**output**”
    1. Presionamos [**OK**] y vemos como ya queda conectado el “**Partern Link**” del “**OSB (Oracle Service Bus)**”



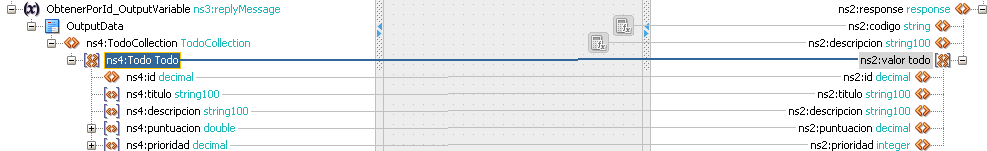
1. Para poder terminar con la conexión hacia el “**Partern Link**” del “**OSB**”; tenemos que realizar dos operaciones de asignación:
   1. La primera es asignar la variable de entrada del proceso, obtenida mediante (), a la variable de entra del “**invoke**” ().
   2. La segunda es un mapeo parecido, pero en este caso es la variable de salida del “**invoke**” () a la variable de salida del proceso ().
2. Comencemos por la primera, en este caso, solo tenemos que realizar un simple proceso de asignación:
   1. Colocamos un nodo de asignación () entre el “**input**” () y el “**invoke**” (). Quedando de la siguiente forma:



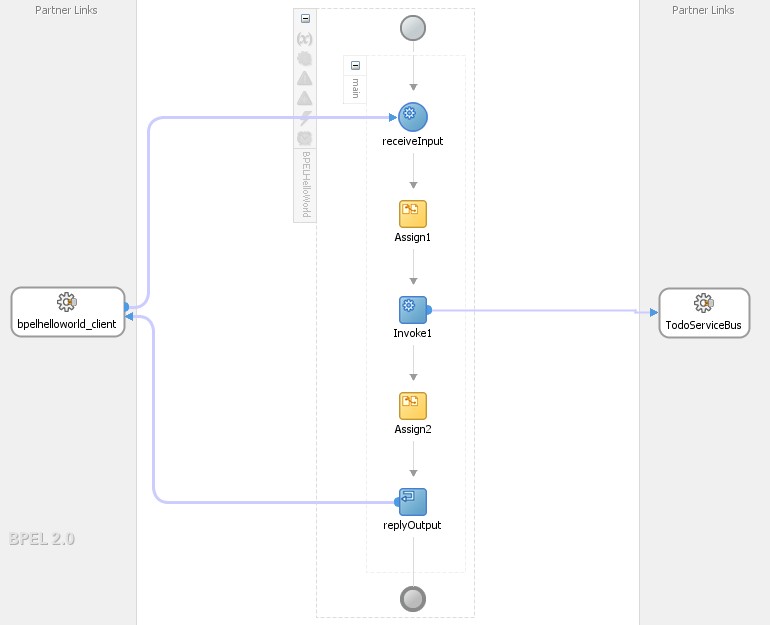
* 1. Ahora solo resta mapear el valor de la entrada del proceso al de la entrada del “**invoke**”. Para ver como se asignan variables, ver el **tutorial 4 – 1.4.1**. Después del proceso, la variable de entrada queda de la siguiente forma:



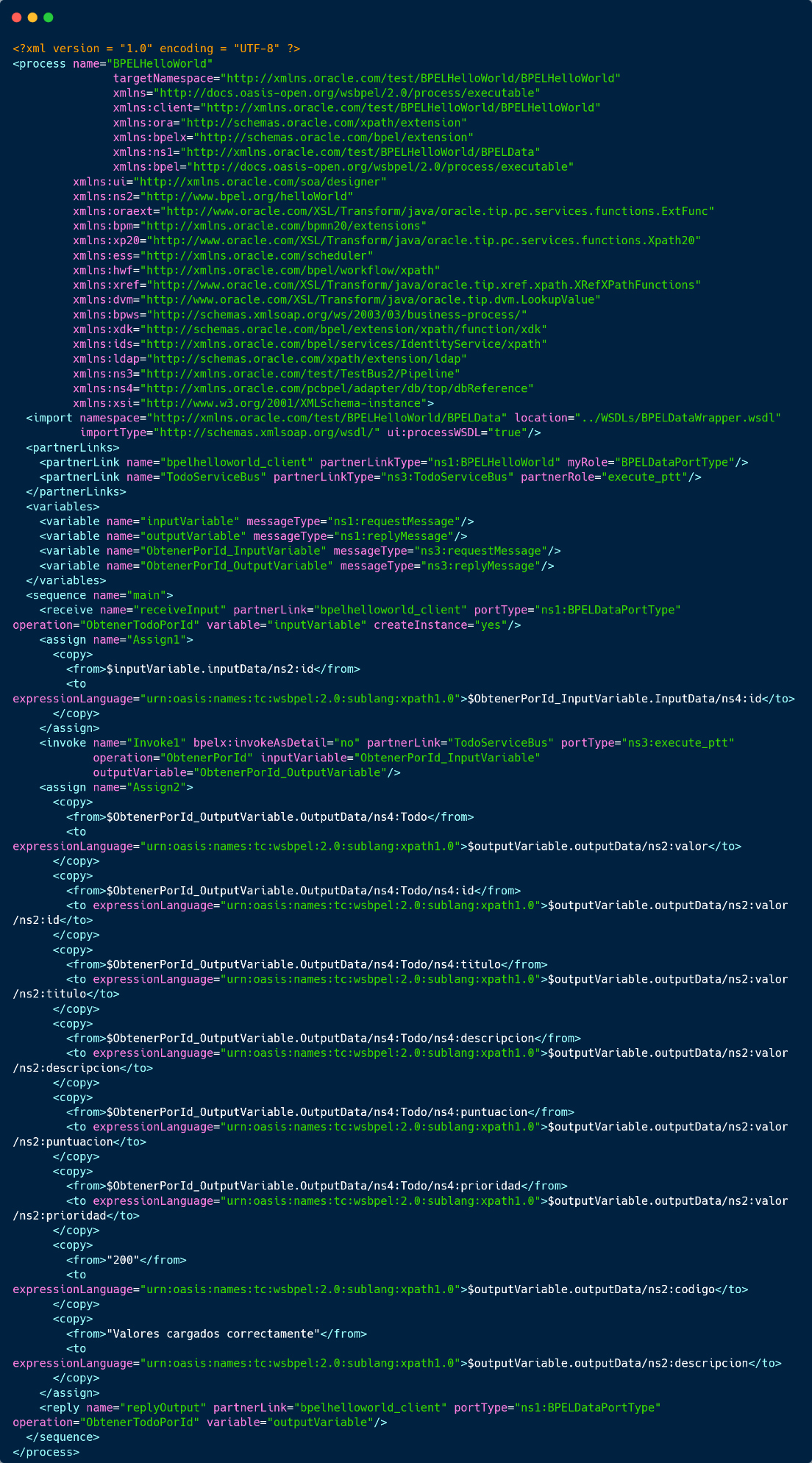
1. Ahora solo nos restaría hacer el mapeo desde la variable de salida del “**invoke**” () a la variable de retorno de nuestro flujo.
   1. Creamos un asignar entre el “**invoke**”y el “**output**” ().
   2. Asignamos los valores correspondientes y completamos con valores literales los campos de “**codigo**” y “**descripcion**”. Para ver como se asignan variables, ver el **tutorial 4 – 1.4.1**.
   3. La asignación queda de la siguiente forma:



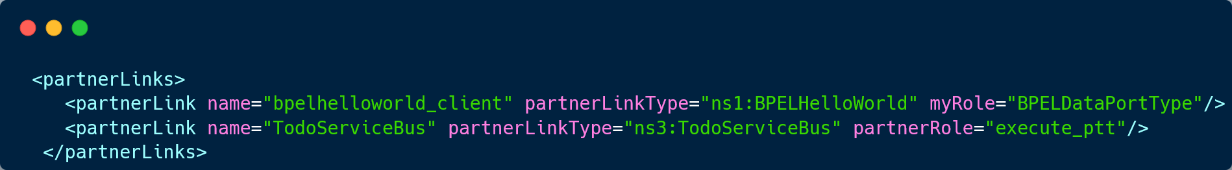
1. Después de completado todo el procedimiento, el flujo queda modelado de la siguiente forma:



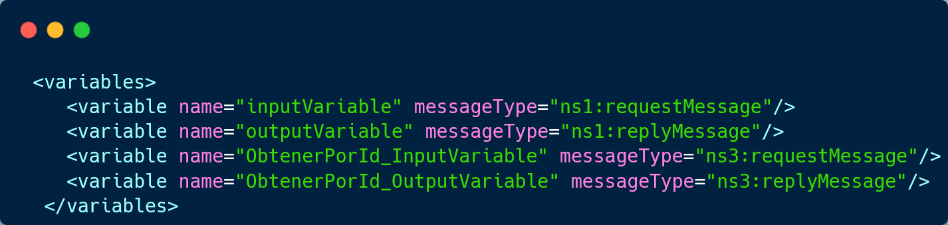
1. Antes de pasar al despliegue y probar. Vamos a echarle un vistazo al código generado por el “**BPEL**”.



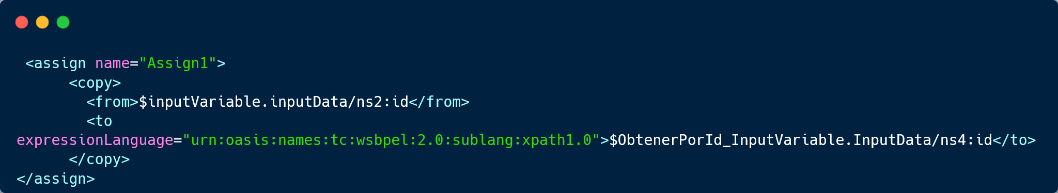
* 1. De ese código podemos ver lo siguiente:
     1. Declaración de los “**partner links**”



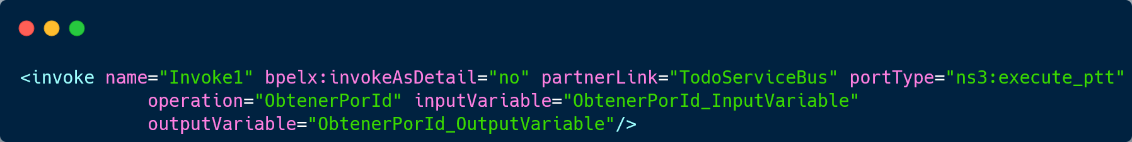
* + 1. Declaración de las variables globales:



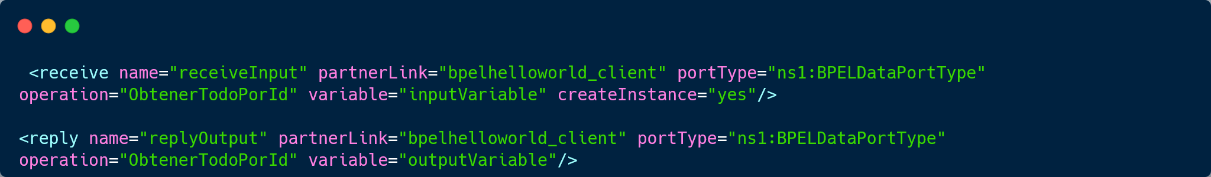
* + 1. La asignación de la variable de entrada del sistema a la variable de entrada del “**invoke**”.
       1. Todas las asignaciones se realizan de la misma forma, por lo que no se va a poner las asignaciones de la variable de salida.



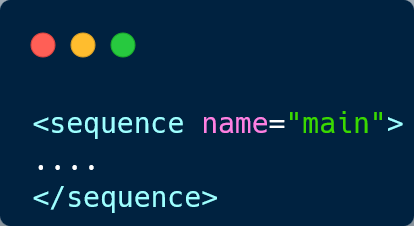
* + 1. La creación de un “**invoke**” es de la siguiente forma:
       1. En este caso es bueno aclarar que las variables se deben crear primero si se va a crear el “**invoke**” desde el código.
       2. De la forma visual, el crea primero las variables y después es que las asigna, aunque no lo parezca a simple vista.



* + 1. La declaración del “**input**” y el “**output**” del proceso.



* + 1. Todo el flujo de datos desde el “**receive**” y el “**reply**” está encerrado dentro de una secuencia:
       1. Las secuencias obligan a que se deba procesar en el mismo orden en que aparecen los nodos.



* 1. Ya con esto tenemos un entendimiento de como se realiza el proceso desde el código.

1. Por último, desplegamos y probamos que funcione el servicio. **Ver tutorial 1 – 5**. Si todo va bien, el resultado de la prueba será el valor de un dato TODO de la BD.